Microlog Microfiche Collection
Resolution Chart

Copyright

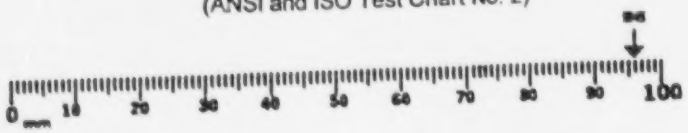
Copyright for the ***contents*** of this microfiche is held by the publisher of the original document. Please contact the original publisher for guidance in all copyright matters. Copyright in this ***microfiche edition*** is held by Micromedia Limited.

Quality of the Microfiche Reproduction

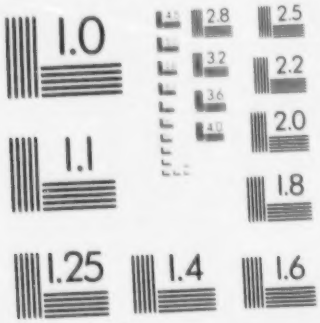
Micromedia Limited maintains the highest standards for micrographic reproduction. Attempts are made to obtain the best quality print version to microfilm. However, there are some printing styles, colours and graphic layout styles (including maps, graphs and tables) which do not reproduce well. Because of the value of the original document, Micromedia has filmed it recognizing that it may not be as easy to read as the original document.

Some pages have been filmed twice to improve the clarity of the reproduction. Consult both reproductions to ensure that all of the original text is read.

Microcopy Resolution Test Chart
National Bureau of Standards
Standard Reference Material 1010a
(ANSI and ISO Test Chart No. 2)



PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT



MS109

NATIONAL MICROGRAPHICS ASSOCIATION

24x

micromedia
a division of IHS Canada

20 Victoria Street, Toronto, ON M5C 2N8
Tel.: (416) 362-5211
Toll free: 1-800-387-2689
Fax: (416) 362-6161

Collection MICROLOG sur Micromédia
Tableau de résolution

Droits d'auteurs

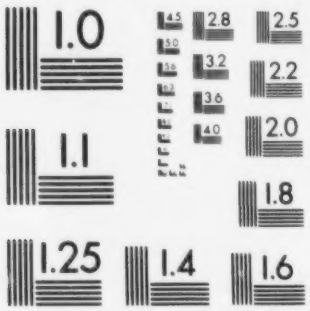
Les droits d'auteurs du conenu de cette microfiche sont détenus par l'éditeur du document original. Veuillez communiquer avec celui-ci pour obtenir des renseignements sur tout ce qui touche les droits d'auteurs. Les droits d'auteurs de cette édition sur microfiche sont détenus par Micromédia Limitée.

Qualité de reproduction sur microfiche

Micromédia Limitée maintient les normes de reproduction micrographique les plus élevées. Elle fait tout son possible pour obtenir la version sur papier à reproduire sur microfiche de la meilleure qualité possible. Cependant, certains styles d'imprimerie, de couleurs et de style de mise en page graphique (incluant des cartes, des grahiques et des tableaux) se reproduisent mal. En raison de la valeur du document original, Micromédia l'a reproduit tout en reconnaissant qu'il ne sera peut-être pas aussi lisable que le document original.

Tableau de résolution micrographique
National Bureau of Standards
Standard Reference Material 1010a
(ANSI and ISO Test Chart No. 2)

PM-1 3½"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT



MS109

24x

NATIONAL MICROGRAPHICS ASSOCIATION

micromedia
a division of IHS Canada

20 Victoria Street, Toronto, ON M5C 2N3
Tel.: (416) 362-5211
Toll free: 1-800-387-2689
Fax: (416) 362-6161

Université de Montréal
Études Commerciales

Université de Montréal
C.P. 6128, Succursale Centre-ville,
Montréal (Québec) Canada H3C 3J7

Téléphone : +1-514-343-7575
Télécopieur (Fax) : +1-514-343-7121
crt@crt.umontreal.ca
www.crt.umontreal.ca/CRT/

CRT-2000-15

Centre de recherche sur les transports (C.R.T.)
Centre for Research on Transportation

AN EXACT ALGORITHM FOR THE ELEMENTARY SHORTEST PATH PROBLEM WITH RESOURCE CONSTRAINTS: APPLICATION TO SOME VEHICLE ROUTING PROBLEMS

by

Cyrille Guegen
Pierre Dejax
Moshe Dror
Dominique Feillet
Michel Gendreau

May 2000

101-01783/

PUBLICATION
CRT-2000-15

*AN EXACT ALGORITHM FOR THE ELEMENTARY SHORTEST PATH PROBLEM WITH RESOURCE CONSTRAINTS: APPLICATION TO SOME VEHICLE ROUTING PROBLEMS**

by

Cyrille Gueguen[1,2]
Pierre Dejax[3]
Moshe Dror[4]
Dominique Feillet[1]
Michel Gendreau[5]

* *This research was supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC).*

[1] Laboratoire Productique Logistique, Ecole Centrale Paris, Grande Voie des Vignes, 92295 Chatenay-Malabry Cedex, FRANCE <feillet@pl.ecp.fr>

[2] Département de Recherche Opérationnelle, Informatique et Télécommunications, AIR FRANCE, 1, av. du Maréchal Devaux, 91551 Paray-Vieille-Poste Cedex, FRANCE <cygueguen@airfrance.fr>

[3] Département automatique et Productique, École des mines de Nantes, 4, rue Alfred Kastler, BP20722, 44307 Nantes Cedex 3, FRANCE <Pierre.Dejax@emn.fr>

[4] College of Business and Public Administration, University of Arizona, Tucson, Arizona 85721, U.S.A. <mdror@mis.arizona.edu>

[5] Département d'informatique et de recherche opérationnelle et Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal (Québec), CANADA H3C 3J7 <michelg@crt.umontreal.ca>

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2000

Abstract In this paper we propose a solution procedure for the Elementary Shortest Path Problem with Resource Constraints (ESPPRC). A relaxed version of this problem in which the path does not have to be elementary has been the backbone of a number of solutions based on column generation procedure for such important problems as vehicle routing, crew-pairing, and others. In many cases relaxing the restriction of an elementary path resulted in optimal solutions in a reasonable computation time. However for a number of other problems such as the Selective Vehicle Routing Problem and the Prize Collecting Vehicle Routing Problem, the elementary path restriction is necessary and its relaxation might result in suboptimal solutions. We propose an exact procedure for the ESPPRC which extends the classical label correcting algorithm originally developed for the relaxed (non elementary) path version of this problem. We present computational experiments of this algorithm for our specific problem and embedded in a column generation scheme for the classical Vehicle Routing Problem with Time Windows.

Key words: shortest path, column generation, vehicle routing.

Résumé Dans cet article nous proposons une procédure de résolution pour le Problème du Plus Court Chemin Elémentaire avec Contraintes de Ressources (ESPPRC). Une version relaxée de ce problème, dans laquelle le chemin n'a pas à être élémentaire, a été l'épine dorsale de nombreuses procédures de résolutions basées sur les techniques de génération de colonnes, notamment pour des problèmes de tournées de véhicules ou d'horaires d'équipes de travail. Dans de nombreux cas, cette relaxation permettait d'obtenir des solutions optimales dans des temps raisonnables. Cependant, pour un certain nombre d'autres problèmes tels que le Problème de Tournées Sélectives et le Problème de Collecte de Gains, il est nécessaire de maintenir les plus court chemins élémentaires sous peine de ne plus garantir l'optimalité. Nous proposons une procédure de résolution exacte pour le ESPPRC qui étend l'algorithme classique de programmation dynamique développé dans le cas de chemins non élémentaires. Nous présentons des expérimentations numériques pour le problème propre, ainsi que pour un algorithme de génération de colonnes faisant appel à lui pour résoudre le Problème de Tournées de Véhicules avec Fenêtres de Temps.

Mots-clés : plus court chemin, génération de colonnes, tournées de véhicules.

1 Introduction

In this paper we propose a solution procedure for the Elementary Shortest Path Problem with Resource Constraints (ESPPRC). A relaxed version of this problem, the Shortest Path Problem with Resource Constraints (SPPRC), in which the path does not have to be elementary has been the backbone of a number of solutions based on column generation procedure. This approach has been successfully applied by Desrochers *et al.* [5], Kohl *et al.* [16] for the Vehicle Routing Problem with Time Windows (VRPTW), by Lavoie *et al.* [17] for the airline crew pairing, by Graves *et al.* [11] for the flight crew scheduling to mention only some examples. In these problems, one might have resource constraints related to time (time windows, route duration), capacity (vehicle capacity) and many other types of resources. For these problems, this column generation approach results in optimal solutions in a reasonable computation time.

An important class of algorithms for solving SPPRC uses a dynamic programming approach. One possibility consists of examining the successors of a node, yielding a reaching algorithm. A second possibility examines the predecessors of a node, generating a pulling algorithm. More methods of the first type have been developed and it is possible to distinguish, in this class, methods which are an extension of the Ford-Bellman's algorithm (which works by label correcting), see Desrochers [4], and methods which are an extension of Dijkstra's algorithm (which works by permanent marking of the labels). Other methods have been studied. Beasley and Christofides [1] have developed a Lagrangean relaxation procedure which gives interesting results. Solving the SPPRC is also closely related to solving multi-objective shortest path problems, which have received the attention of many researchers. Generally, in this kind of problem, no path is optimal for all the objectives simultaneously and it is necessary to generate all the non dominated paths (i.e. Pareto optimal paths). The interested reader may find an exhaustive revue of the multi-objective shortest path problems in Current and Marsh [3]. Usually, the methods developed for these problems only work on graphs which have non-negative lengths (see for example Warburton [19], Hansen *et al.* [13]). We can also mention a two phase algorithm proposed by Jaumard *et al.* [15] for solving the SPPRC but this algorithm requires an acyclic graph.

However, for some vehicle routing applications that we will develop later, the sub-problem encountered in the column generation procedure is necessarily an ESPPRC. To our knowledge, no paper has been dedicated to the ESPPRC when the graph contains negative costs, which frequently happens in the column generation context. It is only possible to mention Gondran

and Minoux [10] who proposed a sub-optimal approach for solving the Elementary Shortest Path Problem (without resource constraint) in a graph that might contain negative cost cycles. Dror [6] proved that the ESPPRC is NP-hard in the strong sense.

The algorithm proposed in this paper is able to optimally solve the ESPPRC in any kind of graph, even if they contain negative costs cycles, which is usually very difficult to apprehend.

In the next section, we describe the process of the column generation technique applied to the VRPTW by Desrochers *et al.* [5]. Indeed, it justifies that it is impossible to always use a SPPRC instead of an ESPPRC as a subproblem in the column generation process, as demonstrated in section 3. In section 4, we extend Desrochers [4] label correcting algorithm for the solution of the ESPPRC. Computational results of section 5 conclude the paper.

2 Shortcomings of the classical column generation solution approach for some vehicle routing problems

The basic VRPTW consists in designing an optimal set of delivery or collection routes from a depot to a number of customers, subject to time windows and capacity constraints.

One of the most successful recent implementations of the column generation solution approach for solving VRPTW was proposed by Desrochers *et al.* [5]. The backbone of this approach is the introduction of the SPPRC as a sub-problem in the column generation procedure.

Formally, let N be the set of customers. Desrochers *et al.* [5] recall the following formulation of the Classical Set Partitioning Model (CSPM) as a master problem for the routing setting:

$$min \sum_{k \in \Omega} c_k x_k$$

$$\sum_{k \in \Omega} a_{ik} x_k = 1, \forall i \in N$$

$$x_k \in \{0,1\}, \forall k \in \Omega$$

where Ω is the set of feasible routes, the indice k represents the routes and a_{ik} is equal to 1 if route $k \in \Omega$ visits customer i and 0 otherwise. This

formulation induces the use of an ESPPRC solution procedure dedicated to the sub-problem.

Instead of the set partitioning model described above, Desrochers *et al.* [5] propose to use a set covering formulation where they replace the " = " of the service constraint by a " $\geq$ ". Their idea is to adapt the classical representation of the a_{ik}s in the CSPM. In their new model, the a_{ik}s are integer and represent the number of times route $k \in \Omega$ visits customer i. The sub-problem associated with this new formulation becomes a SPPRC, and the paths are no longer required to be elementary. Therefore, it allows the implementation of an efficient algorithm to solve this new sub-problem. This relaxation is correct if the triangle inequality is satisfied because, in this case, there exists an optimal solution of the problem that visits each customer at most once.

The ideas of Desrochers *et al.* [5] are computationally very attractive because there exist dynamic programming based procedures which efficiently solve the non elementary path relaxation of the ESPPRC with a pseudo-polynomial worst case complexity.

It is important to note, as mentioned by Gelinas *et al.* [9] that this method leads to a weakening of the lower bound. But they do not provide any computational result. The computational results of section 5.2 will confirm this property.

We will now show that for some classes of vehicle routing problems, the method of Desrochers *et al.* [5] can not be used directly. For instance, for the Selective Vehicle Routing Problem with Time Windows (SVRPTW) and the Prize Collecting Vehicle Routing Problem with Time Windows (PC-VRPTW), detailed in Gueguen [12], profits are associated to customers visits. Consequently, at most one visit is allowed for each customer and a visit is not required for each of them. If we use directly the method of Desrochers *et al.* [5], and if some customers are very profitable, it would be possible to generate routes that visit these customers more that once, which is of course not acceptable. For these problems, it is critical that the sub-problem of the column generation process (see Gueguen [12] for details) is an elementary shortest path problem satisfying the resource constraints.

3 Outlook for solving the ESPPRC

In this section we discuss some straightforward extensions of Desrochers's algorithm [4] to adapt it for the solution of the ESPPRC. We first remind

the notations and principles of Desrochers's algorithm [4]. Then, we show that these modifications unfortunately do not ensure the optimality of the solution. In the next section, we describe an algorithm, also extended from Desrochers's algorithm [4] and using an idea of Beasley and Christofides [1], to solve the ESPPRC to optimality.

3.1 Description of the ESPPRC

Defining the problem more formally, let $G = (N, A)$ be a network where A is the set of arcs and N the set of nodes ($|N| = n$) including an origin node p and a destination node d. A cost c_{ij} and a non negative time duration t_{ij} are associated with each arc $(i, j) \in A$. Casting the problem in a delivery setting, if a vehicle route includes a node $i \in N$, then the vehicle has to deliver to i a commodity of quantity $q_i \geq 0$. In case of delivery time windows for node i, the delivery has to take place within a time frame $[a_i, b_i]$. Let Q be the capacity of the vehicle. Time windows and vehicle capacity are two of the resources considered but it is possible to consider many other resources. Suppose that the number of resources is L. Let d_{ij}^l be the consumption of resource l along arc (i, j). The objective is to construct an elementary path from p to d at minimum cost that satisfies all the resource constraints.

3.2 Notations and principle of Desrochers's algorithm for the SPPRC

Let us recall Desrochers' notations [4] for solving the SPPRC. With each path X_{pj} from the origin p to node j is associated a state $(T_j^1, T_j^2, \ldots, T_j^L)$ corresponding to the quantity of each of the L resources used by the path and a cost $C(T_j^1, T_j^2, \ldots, T_j^L)$. To simplify the notations a label $(T_j^1, T_j^2, \ldots, T_j^L, C_j)$ is defined for each path, where C_j is equal to the cost of the label. This label represents both the state value and the cost of a path X_{pj}. In addition, R_j will be used to represent the state value. We need one more definition from this article. Let X'_{pj} and X^*_{pj} be two distinct paths from p to j with associated labels (R'_j, C'_j) and (R^*_j, C^*_j). Then, according to Desrochers [4], X'_{pj} dominates X^*_{pj} (or $X'_{pj} \prec X^*_{pj}$) if and only if $C'_j \leq C^*_j$ and $T_j^{l'} \leq T_j^{l*}$ for $l = 1, \ldots, L$ and $(R'_j, C'_j) \neq (R^*_j, C^*_j)$.

To obtain the optimal solution of the shortest path problem, at each node j of the graph, one just needs to consider non dominated labels, i.e. non dominated paths. This is an extension of Bellman's optimality principle (Desrochers [4]).

One point of inquiry for our research was to see if one can easily adjust the previous algorithm dedicated to the SPPRC in order to solve the ESPPRC optimally.

The example of Figure 1 illustrates that when considering the dominance relation proposed by Desrochers [4], the optimal elementary path may be dominated during the process.

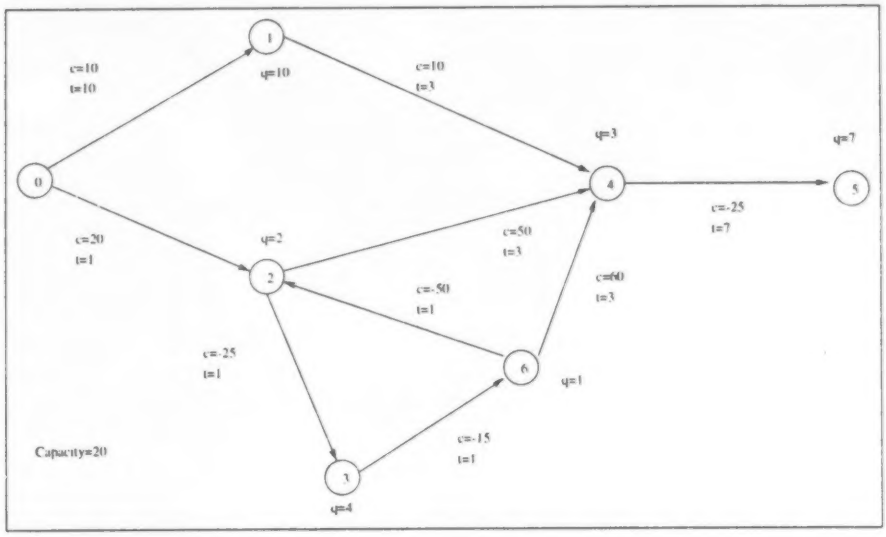


Figure 1: Demand and arc duration.

Here, the considered resources are time and the capacity of the vehicle (Q is equal to 20 and time windows are $[0, 50]$ for each node). Applying the dominance relation, we obtain three labels at node 5:

- label 1 : $(q = 12, t = 11, c = 45)$, corresponding to path $0 \rightarrow 2 \rightarrow 4 \rightarrow 5$,
- label 2 : $(q = 17, t = 13, c = 15)$, corresponding to path $0 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 4 \rightarrow 5$,
- label 3 : $(q = 19, t = 14, c = -45)$, corresponding to path $0 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 2 \rightarrow 4 \rightarrow 5$.

Examining the graph carefully, we find another path going from node 0 to node 5: $0 \rightarrow 1 \rightarrow 4 \rightarrow 5$ $(q = 20, t = 20, c = -5)$. This path is actually the optimal elementary path between nodes 0 and 5, but its label has been dominated by label 3. Thus, seeking the optimal label among whose

corresponding to an elementary path at node 5 will lead to a suboptimal one (label 2).

Claim 1 *One cannot find the optimal elementary path by solving an SPPRC and selecting an optimal path at node d among whose corresponding to an elementary path.*

A second idea for adjusting Desrochers's algorithm [4] to solve the ESPPRC, might be to check that at each step, only elementary paths are generated. We just need to test, each time a label is extended to a node, that this node has not been visited by the path previously.

The example of Figure 2 proves that this idea does not lead to an optimal solution.

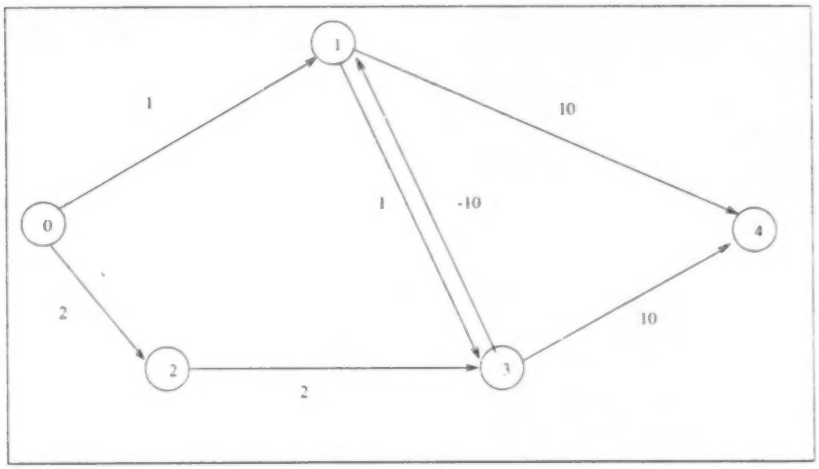


Figure 2: Traveling costs.

At node 3, the label corresponding to path $0 \rightarrow 2 \rightarrow 3$ might be dominated by the label corresponding to path $0 \rightarrow 1 \rightarrow 3$ before being extended to node 1. Then, the optimal elementary path $0 \rightarrow 2 \rightarrow 3 \rightarrow 1 \rightarrow 4$ will not be generated.

Claim 2 *One cannot find the optimal elementary path by solving the SPPRC and checking that only elementary paths are generated during the process.*

We now describe an algorithm able to solve the ESPPRC in graphs that might contain negative cost cycles.

4 A label correcting algorithm to solve the ESPPRC

Beasley and Christofides [1] proposed a procedure for finding elementary paths in a graph. One only has to add an extra resource k for each node k and restrict that resource to the set $\{0,1\}$. When a path visits node k, it consumes resource k. Beasley and Christofides [1] thought that this solution would only be suitable for solving problems for which the number of resources is small and they did not test the method.

In this paper we adapt this idea to Desrochers's Label Correcting algorithm [4] and test it to see if it can generate optimal solutions for problems of moderate size.

4.1 Definitions and claims

As proposed by Beasley and Christofides [1], we add in the definition of labels an extra resource k for each node k. In order to compare labels more efficiently, we also add another resource s_j to each label at each node j, counting the number of nodes previously visited by the path coming to this node. Indeed a label L_1 cannot dominate a label L_2 if it visits more nodes. So, instead of comparing almost all the resources corresponding to the n nodes, a number of times one test will be enough to compare two labels.

Definition 1 *To each path X_{pj} from the origin p to node j, let associate a state*
$R_j = (T_j^1, \ldots, T_j^L, s_j, V_j^1, \ldots, V_j^n)$ *corresponding to the quantity of the resources used by the path, the number of visited nodes, and the visitation vector ($V_j^i = 1$ if the path visits node i, 0 otherwise).*

Note that $s_j = \sum_{i=1}^{n} V_j^i$.

A new dominance relation is stated to compare two labels according to the new definition of labels.

Definition 2 *Let X'_{pj} and X^*_{pj} be two distinct paths from p to j with associated labels (R'_j, C'_j) and (R^*_j, C^*_j). Then X'_{pj} dominates X^*_{pj} (or $(R'_j, C'_j) \prec (R^*_j, C^*_j)$) if and only if $C'_j \leq C^*_j$, $s'_j \leq s^*_j$, $T_j'^l \leq T_j^{*l}$ for $l = 1, \ldots, L$, $V_j'^i \leq V_j^{*i}$ for $i = 1, \ldots, n$, and $(R'_j, C'_j) \neq (R^*_j, C^*_j)$.*

This dominance relation is actually a simple extension of the relation proposed by Desrochers [4] and it still allows the use of the original Desrochers

algorithm. The principle of the algorithm remains the same: all efficient paths are the extension of an efficient path.

We now show that during the algorithm execution, one needs only to consider non-dominated paths. If an efficient path X_{pj} could be the extension of a path X_{pi} (dominated by X'_{pi}) by the arc $(i,j) \in A$, it would be possible to extend X'_{pi} by the same arc to obtain a path X'_{pj} whose label is either dominating or equal to the one of X_{pj}.

Claim 3 *During the execution of the algorithm, we need only to consider non-dominated paths.*

Proof

Let (R'_i, C'_i) and (R_i, C_i) be two labels at node i with $(R'_i, C'_i) \prec (R_i, C_i)$.

Henceforth, we will not remind which path or label we are referring to when even it is obvious.

If it is possible to extend path X_{pi} to j, it is possible to extend path X'_{pi} to j because $V_i'^j \leq V_i^j = 0$.

Then, we have:

- $C'_i + c_{ij} \leq C_i + c_{ij}$,
- $s'_i + 1 \leq s_i + 1$,
- $T_i'^l + d^l_{ij} \leq T^l_i + d^l_{ij}$ for $l = 1, \ldots, L$. As $T_j'^l = max\{a^l_j, T_i'^l + d^l_{ij}\}$ and $T^l_j = max\{a^l_j, T^l_i + d^l_{ij}\}$ we have $T_j'^l \leq T^l_j$ for $l = 1, \ldots, L$,
- $V_j'^k \leq V^k_j$ for $k = 1, \ldots, n$ with $k \neq j$ and $V_j'^j = V^j_j = 1$.

Then, (R'_j, C'_j) is either equal or dominates (R_j, C_j) ⊡

With this new definition of labels, one is likely to keep many more labels for each node because many labels which were dominated in the previous definition of labels will not be dominated with the new one. However, we are sure to keep only labels corresponding to elementary paths and to get the optimal solution.

4.2 Improved definition of labels

To limit the number of labels for each node, we propose a new definition of the labels. We show that, in some cases, it is not useful to consider the resources attached to some nodes. Thus, we act as if extending a label were consuming resources attached to some nodes. This algorithmic modification is computationally attractive because some labels which could not be compared considering all the resources will become comparable and some of them are going to be dominated. The dominance relation will thus become sharper as it is shown in the following example.

Let us consider three nodes and two labels L_1= | 10 | 2 | 1 | 0 | 1 | and L_2= | 5 | 2 | 0 | 1 | 1 | at node 3 where the different columns correspond to the time, the number of visited nodes and the consumption of the resources associated to the nodes 1, 2 and 3. With the present dominance relation, these two labels are not comparable. Let us suppose that the label L_1 can not be extended to node 2 due to the time resource limitation for example. Then, it would be interesting if in our algorithm, L_2 would dominate L_1. The idea is that we can act as if the resource associated to the node 2 has been consumed by the label L_1. Consequently, we propose for the nodes that have not been visited before and that are unreachable by a label (according to one of the resources), to say that the corresponding resource has been consumed by this label. So, with the same example as above, we obtain two new labels: L'_1= | 10 | 3 | 1 | 1 | 1 | and L'_2= | 5 | 2 | 0 | 1 | 1 | . The label L'_2 dominates the label L'_1. If a third label L_3 dominates L_2, then it dominates also L'_2. Finally, if L_2 dominates L_3, then the transformed label L'_2 dominates the label L'_3 because all the transformations that were made on L_2 were also made on L_3.

We now have to formalise this concept. Let us propose a new definition for the labels including the above property.

Definition 3 *With each path X_{pj} from the origin p to node j, associate a state $R_j = (T_j^1, \ldots, T_j^L, s_j, V_j^1, \ldots, V_j^n)$ corresponding to the quantity of the resources used by the path, the number of nodes that are unreachable (including visited nodes) and the visitation vector defined by $V_j^i = 1$ if the path has visited node i or if with the current value of consumption of the different resources, it is impossible to reach this node.*

With the above definitions, we can keep the same dominance relation and only consider non-dominated paths.

Claim 4 *During the execution of the modified algorithm, we need only to consider non-dominated paths.*

Proof
For the clarity of the proof, we just consider the above property applied to time which will be here the first resource of a label.

Consider two labels at node i that are extended to node j such that: $(R'_i, C'_i) \prec (R_i, C_i)$

If it is possible to extend path X_{pi}, it is possible to extend path X'_{pi} because $V_i^{'j} \leq V_i^j = 0$.

We have:
$C'_i + c_{ij} \leq C_i + c_{ij}$ and $T_i^{'l} + d_{ij}^l \leq T_i^l + d_{ij}^l$ for $l = 1, \ldots, L$.
As $T_j^{'l} = max\{a_j^l, T_i^{'l} + d_{ij}^l\}$ and $T_j^l = max\{a_j^l, T_i^l + d_{ij}^l\}$, we have $T_j^{'l} \leq T_j$ for $l = 1, \ldots, L$.

We have still to check the resources associated to the visitation vector:

- For the resource corresponding to node j: $1 = V_j^{'j} \leq V_j^j = 1$.

- For a resource corresponding to a node $k \neq j$:

  - if $T_j^{1'} + d_{jk}^1 \leq b_k^1$ and $T_j^1 + d_{jk}^1 \leq b_k^1$: then it is possible to visit node k, thus, $V_j^{'k} = V_i^{'k}$ and $V_j^k = V_i^k$, and $V_j^{'k} \leq V_j^k$.

  - if $T_j^{1'} + d_{jk}^1 \leq b_k < T_j^1 + d_{jk}^1$: then it is impossible to return to node k with path X_{pj}. We have to set $V_j^k = 1$ and possibly $s_j = s_i + 1$ if node k was not visited by the path X_{pj} before, and we do not change the other values. We are sure that $V_j^{'k} \leq V_j^k$.

  - if $b_k^1 < T_j^{1'} + d_{jk}^1 \leq T_j^1 + d_{jk}^1$: then it is impossible to visit node k: $V_j^{'k} = 1$ and $V_j^k = 1$, and $V_j^{'k} \leq V_j^k$.

For every case and every k, we have $V_j^{'k} \leq V_j^k$. Then

$$\sum_{k=1}^{n} V_j^{'k} \leq \sum_{k=1}^{n} V_j^k \quad \text{i.e.} \quad s_j^{'} \leq s_j$$

$\Rightarrow (R'_j, C'_j)$ is either equal or dominates (R_j, C_j) $\square$

It is possible to implement this label modification in a efficient way increasing only very slightly the number of required operations. A complete

description of the algorithm is given below.

4.3 Description of the algorithm

We need the following notations to describe the algorithm:

- Λ_i: the list of feasible labels at node i,
- $Succ[i]$: the set of successors of node i,
- E: the list of nodes waiting to be treated,
- $Extend(l_i, j)$: function that returns the extended label from label l_i of node i to the node j (the function Extend updates the consumption of all resources and checks that all the resource constraints are satisfied),
- F_{ij}: the set of labels extended from node i to node j,
- $EFF(L)$: the procedure that keeps only non dominated labels in the list of labels L.

We now describe the procedure ESPPRC(p) which determine all the non dominated paths starting from node p to every node of the graph.

```
ESPPRC(p)
 1  Initialization
 2  L_p ← {(0, ..., 0)}
 3  L_i ← ∅  ∀i ≠ p
 4  E = {p}
 5
 6  repeat
 7          Exploration of the successors of a node
 8          Choose i ∈ E
 9          for all j ∈ Succ[i]
10             do F_ij ← ∅
11                for all l_i ∈ Λ_i
12                   do if V_i^j = 0
13                         then F_ij ← F_ij ∪ {Extend (l_i, j)}
14                if Λ_j ≠ EFF [F_ij ∪ Λ_j]
15                  then Λ_j ← EFF [F_ij ∪ Λ_j]
16                       E ← E ∪ {j}
17
18          Reduction of E
19          E ← E\{i}
```

20
21 **until** $E = \emptyset$

It is possible to implement this algorithm in an efficient way in order to reduce the number of operations that have to be performed. The time complexity of this algorithm is strongly related to the structure of the graph, the numbering of the nodes, and the tightness of resource constraints. If the problem is highly constrained, it is possible to solve quite large problems as it is shown in section 5.1.

4.4 Extension

This method can be adapted for the determination of a path that visits at most one node of some subsets of nodes (i.e. generalized shortest path problem in the flavor of the generalized traveling salesman problem). Instead of having one resource for each node, one has one resource for each subset. When a path visits one node of a given subset, the entire resource of this subset is consumed and is not available to any other node of the subset.

5 Computational results

The computational study is divided into two parts. First, we conducted experiments showing that it is possible to solve ESPPRC of moderate size with the algorithm described in section 4 and showing that the speeding up we obtained with the modification described in section 4.2 is very significant. In the second part, we give some results on the so-called Solomon's [18] problems and show that it is possible to use this algorithm to solve also VRPTW problems by a column generation approach, even when an ESPPRC is not required.

For both parts, we use Solomon's [18] problems which are classified in three categories: the r-problems where the customers are located randomly, the c-problems where the customers are located in clusters and the rc-problems with some random and some clustered structures. The number of customers is 100 for every problem but smaller problems are created by considering only the first 25 or the first 50 customers.

Costs and travel times are calculated with one decimal point and truncation. So, if (x_i, y_i) and (x_j, y_j) denote the coordinates of two customers i and j, then $t_{ij} = st_i + \frac{\lfloor 10\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \rfloor}{10}$ where st_i represents the service time of customer i and $\lfloor x \rfloor$ returns the largest integer smaller than x. Thus, we

are able to compare our results with those of Kohl *et al.* [16] who consider a SPPRC for the solution of the VRPTW and who present the best results known. It is interesting to note that working with non truncated values would not have affected our computational times.

Remark 1 *It is important to recall that the more constrained the problem is, the easier it is to find the optimal solution. For the Solomon's [18] problems, there are only two resource constraints: the time windows and the capacity of the vehicle. If the vehicle capacity is the same for all the data sets, the time windows are very different.*

Computational experiments were carried out on a PC with a 333 MHz processor with 128 Megabytes of RAM. Algorithms were implemented in C++ with Visual C++ 5.0 Compiler. The computational times are calculated with the "*clock()*" function.

5.1 Using the algorithm to solve an ESPPRC

We use Solomon's [18] data sets for the location of the customers, their time windows and the quantity that has to be delivered to these customers in case of visit. Concerning the travel costs, we try to reproduce situations encountered when using the usual column generation schemes. We have decided to generate a variable α_j which was uniformly distributed on the interval $[0, 20]$. The costs are then defined as follows: $c_{ij} = d_{ij} - \alpha_j$, generating negative costs for some arcs.

The geographical distribution of the customers in the c and rc-problems and the costs definition makes them more difficult to solve because of their higher probability of having arcs with a negative cost.

In each problem, we look for the shortest paths starting from the depot and coming back to an artificial copy of the depot. The number of nodes of the problems is thus equal to the number of customers plus two.

For each problem, we give the number of arcs in the graph, its density, the percentage of arcs with a negative cost, the number of non dominated paths at the artificial depot and the time needed to find the solution before and after the improvement described in section 4.2. The last column gives the time without the improvement divided by the time with the improvement. The results are summarized in the Tables 1, 2 and 3. The maximum allowed time to find the solution is 300 seconds.

Problem	♯ Arcs	Density	% neg arcs	Before Imp		After Imp		Gain
				CPU1 (s)	♯ labels	CPU2 (s)	♯ labels	
r101.25	225	34.61	6.67	0.01	28	0.01	12	1
r101.50	813	31.88	7.99	0.04	83	0.03	11	1.3
r101.100	3244	32.12	9.83	11.29	2606	0.41	29	27.5
r102.25	413	63.54	7.74	0.01	37	0.02	22	0.5
r102.50	1504	58.98	7.05	0.68	317	0.22	42	3.1
r102.100	5813	57.55	9.58			280.35	1472	∞
r103.25	525	80.77	6.48	0.03	38	0.03	26	1
r103.50	1970	77.25	6.39	9.16	1221	1.01	101	9.1
r103.100	7705	76.29	8.72					
r104.25	580	89.23	6.21	0.03	41	0.05	32	0.6
r104.50	2398	94.4	5.92			58.96	1121	∞
r104.100	9187	90.96	8.13					
r105.25	301	46.31	5.98	0.01	301	0.01	12	1
r105.50	1068	41.88	7.4	0.18	146	0.09	17	2
r105.100	4261	42.19	9.72			1.63	84	∞
r106.25	462	71.08	7.14	0.02	462	0.03	22	0.7
r106.50	1696	66.51	6.84	3.68	832	0.49	84	7.5
r106.100	6559	64.94	9.5					
r107.25	550	84.61	6.18	0.03	40	0.04	26	0.7
r107.50	2092	82.04	6.12	38.43	2714	1.86	163	20.7
r107.100	8206	81.25	8.7					
r108.25	598	92	6.02	0.04	598	0.05	33	0.8
r108.50	2437	95.57	5.87			67	1196	∞
r108.100	9411	93.18	8.22					
r109.25	411	63.23	6.33	0.02	35	0.02	16	1
r109.50	1534	60.16	6.91	1.20	362	0.29	35	4.1
r109.100	6033	59.73	9.56			11.3	234	∞
r110.25	527	81.08	6.26	0.04	42	0.04	22	1
r110.50	2019	79.18	6.34	54.48	2919	1.17	119	46.6
r110.100	7966	78.87	8.91					
r111.25	526	80.92	6.46	0.03	39	0.03	26	1
r111.50	2035	79.80	6.39	29.97	2356	1.51	125	19.8
r111.100	7958	78.79	8.86					
r112.25	647	99.54	5.72	0.08	45	0.07	41	1.1
r112.50	2528	99.14	5.74			5.24	305	∞
r112.100	10014	99.15	8.05					

Table 1: Solution of the ESPPRC for r-problems.

Problem	♯ Arcs	Density	% neg arcs	Before Imp		After Imp		Gain
				CPU1 (s)	♯ labels	CPU2 (s)	♯ labels	
c101.25	333	51.23	23.12	2.08	2206	0.02	51	104
c101.50	1218	47.76	14.94	154.54	9606	0.13	49	1188
c101.100	4516	44.71	8.28			0.86	75	∞
c102.25	475	73.08	24	88.94	6357	0.88	498	101.1
c102.50	1749	68.59	14.58			1.85	185	∞
c102.100	6684	66.18	7.54			99.04	456	∞
c103.25	570	87.69	21.05			6.46	560	∞
c103.50	2188	85.8	13.76			96.42	1301	∞
c103.100	8526	84.42	7.45					
c104.25	605	93.08	21.32			186.13	1837	∞
c104.50	2462	96.55	13.61					
c104.100	9597	94.96	7.31					
c105.25	358	55.08	21.79	2.31	2206	0.03	52	77
c105.50	1334	52.31	14.09			0.18	50	∞
c105.100	5052	50.02	7.62			1.15	77	∞
c106.25	337	51.85	22.85	2.06	2203	0.03	50	68.7
c106.50	1273	49.92	14.69	255.69	10625	0.16	56	1598.1
c106.100	5423	53.69	7.78			1.7	101	∞
c107.25	378	58.15	20.9	4.02	2698	0.03	58	134
c107.50	1444	56.63	13.09			0.22	65	∞
c107.100	6694	66.28	46.34			125.47	260	∞
c108.25	432	66.46	21.99	10.36	2816	0.08	85	129.5
c108.50	1643	64.43	14			0.45	102	∞
c108.100	6343	62.8	7.5			3.31	168	∞
c109.25	483	74.31	21.95	31.17	4134	0.19	138	164.0
c109.50	1896	74.35	13.61			1.5	227	∞
c109.100	7455	73.81	7.23			11.36	470	∞

Table 2: Solution of the ESPPRC for c-problems.

Problem	♯ Arcs	Density	% neg arcs	Before Imp		After Imp		Gain
				CPU1 (s)	♯ labels	CPU2 (s)	♯ labels	
rc101.25	277	42.62	19.5	0.03	147	0.02	18	1.5
rc101.50	833	32.67	18.85	0.68	748	0.08	29	8.5
rc101.100	3642	36.06	7.93	18.30	2584	0.74	35	24.7
rc102.25	440	67.69	18.18	0.14	237	0.05	54	2.8
rc102.50	1463	57.37	14.49	10.74	1390	0.34	27	31.6
rc102.100	6002	59.43	6.53			8.96	148	∞
rc103.25	546	84	16.12	0.2	257	0.11	74	1.8
rc103.50	1941	76.12	12.57	66.44	1842	1.29	43	51.5
rc103.100	7820	77.43	5.83			242.92	1258	∞
rc104.25	592	91.08	16.05	0.67	312	0.23	116	2.9
rc104.50	2376	93.18	11.7			12.30	409	∞
rc104.100	9274	91.82	5.43					
rc105.25	371	57.08	19.95	0.06	185	0.03	22	2
rc105.50	1242	48.71	16.43	3.09	1155	0.28	28	11
rc105.100	5250	51.98	6.89	248.68	8700	8.54	75	29.1
rc106.25	388	59.69	18.56	0.08	277	0.04	34	2
rc106.50	1243	48.74	17.94	5.03	1606	0.33	31	15.2
rc106.100	5390	53.37	7.33			3.44	65	∞
rc107.25	514	79.08	18.09	0.44	278	0.14	45	3.1
rc107.50	1835	71.96	14.55	65.52	3232	2.51	111	26.1
rc107.100	7542	74.67	6.38			26.97	177	∞
rc108.25	617	94.92	15.88	1.01	307	0.34	92	3
rc108.50	2317	90.86	12.13			8.79	184	∞
rc108.100	9313	92.21	5.52			207.08	486	∞

Table 3: Solution of the ESPPRC for rc-problems.

Our algorithm with the modification succeeds in solving 75 problems on the 87 tested problems in less than 300 seconds while the version without the modification succeeds in only 47. Besides the fact that our modification allows to solve 28 problems more, the most important result lies in the significant reductions of the calculation time that are summarized in Table 4. The first column gives the problem type, the second the number of problems that were solved by both algorithms. Columns 3 and 4 give respectively the mean calculation times before and after improvement. The last column indicates the average of CPU1 divided by CPU2 for each solved problem.

Problem type	♯ problems	Av. CPU1 (s)	Av. CPU2 (s)	Av. Gain
r-problems	22	6.79	0.34	6.92
c-problems	9	61.24	0.17	396.1
rc-problems	16	26.32	0.94	13.6

Table 4: Synthesis of average calculation time and gains.

If we focus on the problems solved by both algorithms, we can see that the average CPU times decrease respectively from 6.79 to 0.34 seconds for the r-problems, from 61.24 to 0.17 seconds for the c-problems and from 26.32 to 0.94 seconds for the rc-problems. The reduction factors are extremely high. Moreover, if we only consider problems solved before the improvement in more than 10 seconds, which implies that the time of solution is mainly affected by the ESPPRC solution procedure, we can see that the gains become respectively 28.65 for the r-problems, 636.14 for the c-problems and 32.60 for the rc-problems.

To conclude this first part of experimental tests, we show on Figure 3 the number of solved problems as a function of time by both algorithms: in full line without the improvement and in dotted line with the improvement. The improved algorithm succeeds in solving 44 problems in less than one second and 54 in less than two seconds while the original algorithm was able to solve only 25 problems in less than two seconds.

These tests prove that if a problem is "sufficiently" constrained, then it is possible to optimally solve problems with 100 nodes in reasonable time and especially that the improvement we proposed is very efficient.

5.2 Using the algorithm inside a column generation solution procedure

Although our algorithm was specifically designed for solving the ESPPRC, we thought it would be interesting to see if it was able to solve some Solomon's

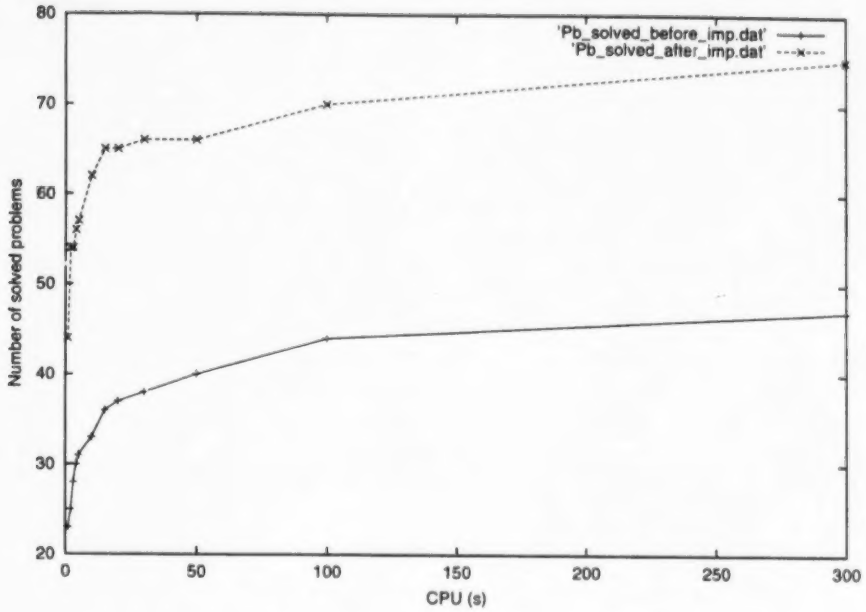


Figure 3: Number of solved problems in function of time.

[18] VRPTW instances when an SPPRC algorithm only is required. Because our objective was just to test the shortest path algorithm, i.e. the subproblem of the well-known column generation scheme, we decided to solve only the linear relaxation of all the instances and we did not try to get the integer optimal solution using a Branch & Price. However, some relaxed solutions are integer and are labeled with an asterisk in the results tables.

LPToolKit version 3.6 [8], a library of C++ classes, was used to generate the master problem for the column generation. This model was solved using the solver CPLEX version 6.0 [14].

For each problem, we give the lower bound $LB1$ found by Kohl *et al.* [16], i.e. the bound obtained before they started to add 2-path cuts strengthening the linear relaxation of the problem. Column 4 indicates the optimal solution OPT found by our algorithm for the relaxed problem. Column 5 gives the gap between these two values. We then give the computational time of our algorithm, the number of generated columns and the number of subproblems solved during the column generation process. Finally, we report in column 3 the total time (to get the optimal integer solution) of Kohl *et al.* [16] algorithm, only when we obtain an integer solution, i.e. when this time is meaningful.

For these tests and to be able to compare the values with those of Kohl *et al.* [16], we recall that we use cost and travel time calculated with one decimal. The results are summarized in tables 5, 6 and 7.

Remark 2 *The tests proposed by Kohl et al. [16] were carried out on a HP9000/829 computer with a 100 Mhz PA7200 processor.*

Remark 3 *To start the column generation process, we developed an adaptation of the savings algorithm of Clarke and Wright [2] which incorporates the time windows constraints to give a feasible solution.*

Remark 4 *At each iteration, all the columns with a negative reduced cost were incorporated into the master problem.*

Our experimentations and results show that our method is promising even for solving the basic VRPTW because we could solve most of the instances solved by Kohl *et al.* [16]. For some of them, we have even calculated bounds which were not found before.

However, our computational running times are higher than those obtained by Kohl *et al.* [16]. This is due to the fact that our algorithm is not designed for solving these problems and that our implementation is limited so far to a basic version of the label correcting algorithm. It does not include all the sophistications of Desrochers's algorithm that were included in the Kohl *et al.* [16] implementation.

These results allow also to measure the gap between the optimal solution of the relaxed problem solved by Desrochers *et al.* [5] or Kohl *et al.* [16] and the optimal solution of the CSPM. We can see that this gap is very often low, most of the time lower than 1% and in fact is null 37 times out of 58. Thus, the improvement is slight but it sometimes leads to reach directly the integer optimal solution (9 times on 87). Moreover, these results show that the method of Desrochers *et al.* [5] only weakened slightly the relaxation.

Finally, we can note that we found two solutions better than those proposed by Kohl *et al.* [16] for problems c108 and c109 in which a misprint should have been made.

Problem	LB1	CPU1 (s)	OPT	Gap (%)	CPU2 (s)	♯ columns	♯ ss-pb
r101.25	*617.100	0.23	*617.100	0	0.86	126	12
r101.50	1043.367		1043.367	0	2.04	391	19
r101.100	1631.150		1631.150	0	19.57	1466	38
r102.25	546.333		546.333	0	1.95	552	15
r102.50	*909.000	3.66	*909.000	0	12.83	1203	20
r102.100	1466.600		1466.600	0	297.58	4526	42
r103.25	*454.600	0.90	*454.600	0	2.57	485	16
r103.50	765.950		769.233	0.42	48.72	2449	25
r103.100							
r104.25	*416.900	1.34	*416.900	0	4.97	842	17
r104.50	616.500		619.077	0.42	1123.09	8739	30
r104.100							
r105.25	*530.500	0.32	*530.500	0	1.21	176	14
r105.50	892.120		892.120	0	5.79	683	25
r105.100	1346.142		1346.142	0	79.14	3059	50
r106.25	457.300		457.300	0	2.50	575	18
r106.50	791.367		791.367	0	20.15	1704	26
r106.100			1226.908				
r107.25	422.925	1.46	*424.300	0.32	3.99	632	20
r107.50	704.438		707.260	0.40	91.45	3222	23
r107.100							
r108.25	396.139		396.820	0.17	6.28	920	17
r108.50			594.699		1627.87	9082	30
r108.100							
r109.25	*441.300	0.40	*441.300	0	2.00	328	19
r109.50	775.096		775.342	0.03	13.00	1109	27
r109.100			1134.281		289.61	4977	47
r110.25	437.300		438.350	0.24	2.33	443	17
r110.50	692.577		695.061	0.36	34.16	1853	30
r110.100							
r111.25	423.788		427.283	0.82	2.83	464	16
r111.50			696.285		73.97	2718	26
r111.100			1055.56				
r112.25	384.200		387.050	0.74	5.29	709	19
r112.50			614.851		242.28	3734	33
r112.100							

Table 5: Solution of the VRPTW by column generation for r-problems.

Problem	LB1	CPU1 (s)	OPT	Gap (%)	CPU2 (s)	♯ columns	♯ ss-pb
c101.25	*191.30	0.56	*191.30	0	3.94	772	38
c101.50	*362.40	2.48	*362.40	0	12.51	1753	60
c101.100	*827.30	5.86	*827.30	0	81.23	2596	89
c102.25	*190.30	1.94	*190.30	0	18.13	3265	24
c102.50	*361.40	13.99	*361.40	0	150.69	7135	36
c102.100	827.30						
c103.25	*190.30	4.03	*190.30	0	122.90	6214	22
c103.50	361.40						
c103.100	826.30						
c104.25	*186.90	8.45	*186.90	0	457.07	1058	27
c104.50	357.25						
c104.100	822.90						
c105.25	*191.30	0.64	*191.30	0	3.14	648	28
c105.50	*362.40	5.83	*362.40	0	16.76	1972	68
c105.100	827.30						
c106.25	*191.30	0.62	*191.30	0	4.61	844	44
c106.50	*362.40	1.25	*362.40	0	11.68	1556	51
c106.100	*827.30	23.70	*827.30	0	165.66	4303	90
c107.25	*191.30	0.61	*191.30	0	3.78	849	33
c107.50	*362.40	3.85	*362.40	0	17.60	2049	58
c107.100	827.30						
c108.25	*191.30	0.64	*191.30	0	4.07	876	30
c108.50	*362.40	8.21	*362.40	0	26.10	2453	53
c108.100	827.30	42.20	*823.80	♯	259.97	4857	79
c109.25	*191.30	0.87	*191.30	0	6.65	1613	29
c109.50	*362.40	5.34	*362.40	0	59.65	3333	52
c109.100	825.64	72.90	*823.80	♯	775.36	10167	90

Table 6: Solution of the VRPTW by column generation for c-problems.

Problem	LB1	CPU1 (s)	OPT	Gap (%)	CPU2 (s)	♯ columns	♯ ss-pb
rc101.25	406.625		406.625	0	2.03	359	23
rc101.50	850.021		850.021	0	5.03	748	32
rc101.100	1584.094						
rc102.25	*351.800	0.55	*351.800	0	3.40	418	29
rc102.50	719.902		721.815	0.26	15.38	957	36
rc102.100							
rc103.25	332.050	3.99	*332.800	0.22	4.76	506	23
rc103.50	643.133		645.281	0.33	76.95	1321	35
rc103.100							
rc104.25	305.825	6.89	*306.600	0.22	15.52	590	24
rc104.50	541.800	169.02	*545.800	0.73	386.38	1691	45
rc104.100							
rc105.25	410.950	3.11	*411.300	0.08	2.78	357	23
rc105.50	754.443		761.558	0.94	15.23	856	35
rc105.100	1471.160						
rc106.25	342.829	7.66	*345.500	0.78	2.87	444	24
rc106.50	664.433		664.433	0	19.86	1155	45
rc106.100							
rc107.25	*298.300	0.73	*298.300	0	5.58	353	27
rc107.50	591.476		603.583	2.04	109.29	1406	41
rc107.100							
rc108.25	293.791	4.28	*294.500	0.24	24.93	590	28
rc108.50	538.957		541.167	0.41	677.67	1633	51
rc108.100							

Table 7: Solution of the VRPTW by column generation for rc-problems.

Before concluding this experimental part, we thought that it was interesting to examine the evolution of the objective function of the relaxed problem against time. We plotted this curve for eight problems with one hundred customers: r101, r102, r105, r109, c101, c106, c108 and c109 (see Figure 4 to 11). With each point of this curve corresponds the end of a complete column generation iteration which is made up of the update of the costs in the sub-problem graph, the solution of the sub-problem, the addition of the new columns in the master problem and the solution of the linear relaxation of the master problem, seeing that most of this time is spent in the solution of the sub-problem.

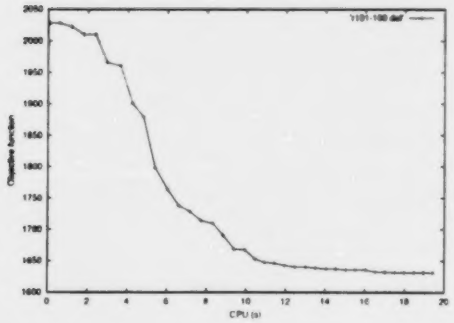


Figure 4: Objective function against time for the solution of problem r101.

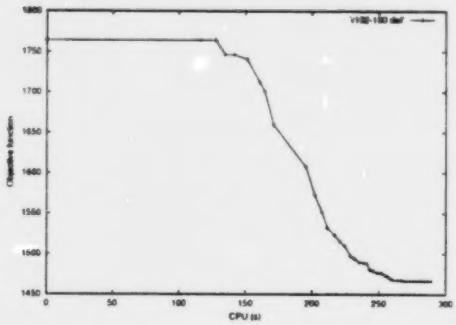


Figure 5: Objective function against time for the solution of problem r102.

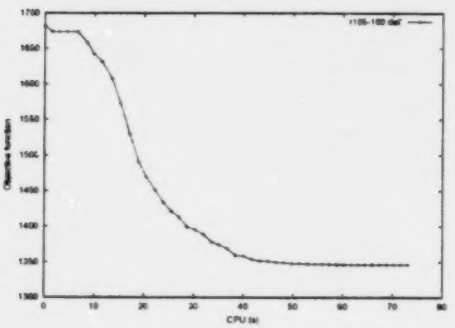


Figure 6: Objective function against time for the solution of problem r105.

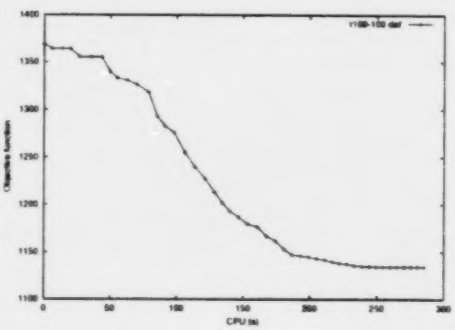


Figure 7: Objective function against time for the solution of problem r109.

These curves are interesting because they bring out a similar behaviour of the evolution of the objective function for the eight problems. It is in fact possible to distinguish three stages in this evolution. During the first stage,

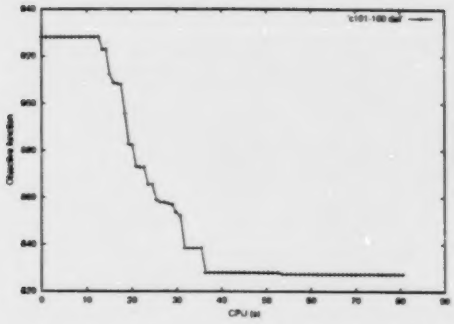


Figure 8: Objective function against time for the solution of problem c101.

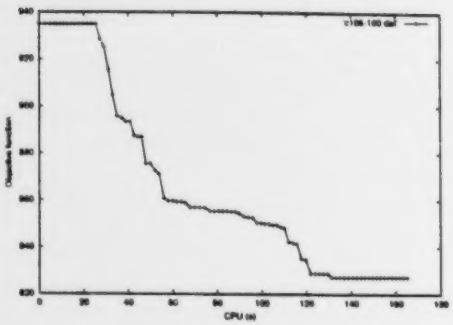


Figure 9: Objective function against time for the solution of problem c106.

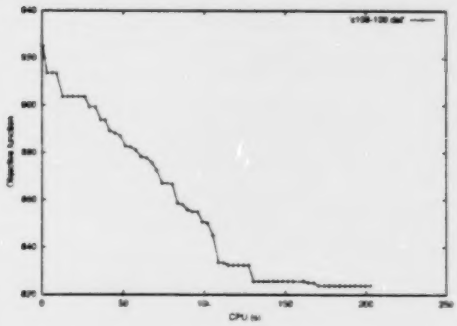


Figure 10: Objective function against time for the solution of problem c108.

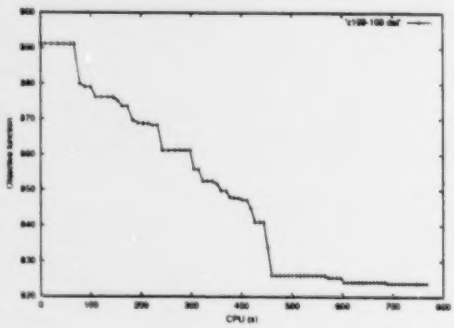


Figure 11: Objective function against time for the solution of problem c109.

the objective function evolves very smoothly at each iteration. Then, during the second stage, each iteration improves in a significant way the objective function. Finally, the third phase is identical to the first one with small variations of the objective function, which shows the difficulty of coming closer to the optimal value of the problem and proving that this value is really the optimal one.

Du Merle *et al.* [7] observed that the convergence of the column generation process is often slow. This statement is verified here, especially for the c-problems, where the customers are located in clusters.

It is also interesting to remark, especially for problem r102 (see Figure 5), that the first iterations take a longer time than those of the rest of the solution.

Finally, it is noteworthy to mention that our algorithm has been implemented in the work of Witucki [20] for the Multiperiodic Vehicle Routing Problem with Dynamic Demand. In Witucki's problems, there are no time windows but the capacity of the vehicles is very restrictive. Some large problems with 80 customers on a 5 days horizon have been solved in reasonable time.

We have also successfully experimented our algorithm for solving Vehicle Routing Problems with Profits (SVRPTW and PC-VRPTW), where the subproblems are elementary shortest path problems. See Gueguen [12] for a complete description of these applications.

6 Conclusion

In this paper we have presented an algorithm for the optimal solution of the Elementary Shortest Path Problem with Resource Constraints (ESPPRC). Such an algorithm is essential for solving special cases of vehicle routing problems such as the Selective and the Prize Collecting Vehicle Routing Problems with a set packing formulation leading to a column generation solution methodology where the shortest path is repeatedly solved as a sub-problem.

Our computational results on a variety of tests problems demonstrate the efficiency of our algorithm and that our approach can be successfully used for solving ESPPRC of moderate sizes. We have also obtained satisfactory results when applying our algorithm to the solution of the Vehicle Routing Problem with Time Windows when an elementary shortest path is not required in the sub-problem. Our approach has also proved to be effective for the solution of the Selective Vehicle Routing Problem with Time Windows, the Prize Collecting Problem with Time Windows and the Multiperiodic Vehicle Routing Problem with Dynamic Demand.

7 Acknowledgments

This work was done while Cyrille Gueguen was member of Laboratoire Productique Logistique of Ecole Centrale Paris. The authors wish to thank Maciej Witucki for numerous suggestions when applying our algorithm to his own problem.

References

[1] BEASLEY J.E. and CHRISTOFIDES N. An algorithm for the resource constrained shortest path problem. *Networks*, 19:379–394, 1989.

[2] CLARKE R.M. and WRIGHT J.W. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12:568–581, 1964.

[3] CURRENT J. and MARSH M. Multiobjective transportation network design and routing problems: Taxonomy and annotation. *European Journal of Operational Research*, 65:4–19, 1993.

[4] DESROCHERS M. An algorithm for the shortest path problem with resource constraints. Technical Report G-88-27, GERAD, 1988.

[5] DESROCHERS M., DESROSIERS J., and SOLOMON M. A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, 40(2):342–354, 1992.

[6] DROR M. Note on the complexity of the shortest path models for column generation in VRPTW. *Operations Research*, 42(5):977–978, 1994.

[7] DU MERLE O., VILLENEUVE D., DESROSIERS J., and HANSEN P. Stabilized column generation. *Discrete Mathematics*, 194(1-3):229–237, 1999.

[8] EURODÉCISION. *LPToolKit, User's Guide and Technical Reference*, 3.6 edition, 1998.

[9] GELINAS S., DESROCHERS M., DESROSIERS J., and SOLOMON M.M. A new branching strategy for time constrained routing problems with application to backhauling. *Annals of Operations Research*, 61:91–109, 1995.

[10] GONDRAN M. and MINOUX M. *Graphes et Algorithmes*. Eyrolles, 1985.

[11] GRAVES G.W., MCBRIDE R.D., GERSHKOFF I., ANDERSON D., and MAHIDHARA D. Flight crew scheduling. *Management Science*, 39(6):657–682, 1993.

[12] GUEGUEN C. *Méthodes de résolution exacte pour les problèmes de tournées de véhicules*. PhD thesis, Laboratoire Productique Logistique, Ecole Centrale Paris, 1999.

[13] HANSEN P., JAUMARD B., and VOVOR T. Solving the bicriterion shortest path problem from both ends. Technical Report G-98-17, GERAD, 1998.

[14] ILOG Inc., CPLEX Division, Incline Village. *ILOG, Using the CPLEX Callable Library*, 6.0 edition, 1998.

[15] JAUMARD B., SEMET F., and VOVOR T. A two-phased resource constrained shortest path algorithm for acyclic graphs. Technical Report G-96-48, GERAD, 1996. To appear in Networks.

[16] KOHL N., DESROSIERS J., MADSEN O.B.G., SOLOMON M.M., and SOUMIS F. 2-path cuts for the vehicle routing problem with time windows. *Transportation Science*, 33(1):101–116, 1999.

[17] LAVOIE S., MINOUX M., and ODIER E. A new approach for crew pairing problems by column generation with an application to air transportation. *European Journal of Operational Research*, 35:45–58, 1988.

[18] SOLOMON M.M. *Vehicle Routing and Scheduling with Time Window Constraints: Models and Algorithms.* PhD thesis, Department of Decision Sciences, University of Pennsylvania, 1983.

[19] WARBURTON A. Approximation of pareto optima in multiple-objective, shortest-path problems. *Operations Research*, 35(1):70–79, 1987.

[20] WITUCKI M. *Problème de tournées de véhicules multi-périodiques avec demande dynamique, Thesis in progress.* PhD thesis, Laboratoire Productique Logistique, Ecole Centrale Paris, 2000.